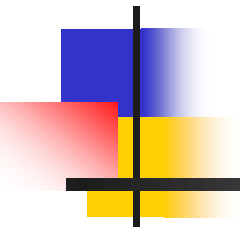


Programski prevodioci 1

Lekcija 3 – Konačni procesori

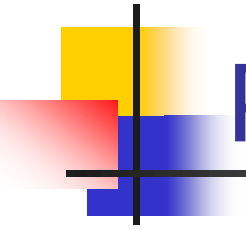
- Prepoznavači i procesori
- Implementacija konačnih automata

Konačni procesori

- Prepoznavač : binarna odluka – ulazna sekvenca se prihvata ili odbija
- Procesor – pored prepoznavanja ulazne sekvence vrši i neku dodatnu obradu kao bočni efekat,
- Primeri dodatne obrade u kontekstu prevođenja programa:
  - Izračunavanje binarne vrednosti konstante
  - Smeštanje simbola u tabelu simbola

Konstrukcija konačnog procesora

- Prepoznavač pretvaramo u procesor na taj način što svakom (nepraznom) ulazu u tabeli prelaza pridružujemo AKCIJU (proizvoljnu sekvencu programskog koda).
- Ulaznu azbuku automata možemo proširiti simbolom kraja ulazne sekvence (end marker) —| koji se javlja jednom u ulaznoj sekvenci, uvek i isključivo na njenom kraju. Praktični razlozi.
- Praznim ulazima, po potrebi, možemo pridružiti akcije za detekciju i oporavak od leksičkih grešaka.



Primer

- Treba konstruisati deterministički procesor za izračunavanje vrednosti neoznačenih realnih konstanti.
- Primeri dozvoljenih konstanti su: 1257,.0392, 1E + 534, 1., 1.E5, 2.0E10
- a nedozvoljenih -21, .E3, E, E10, 12E.

Primer...

- Procesor izračunava binarnu vrednost konstante postavljajući sadržaje sledeća četiri registra:
- NR (number register) sadrži binarnu vrednost celog broja koji se dobija uklanjanjem decimalne tačke iz mantise;
- ER (exponent register) sadrži binarnu vrednost eksponenta koji je naveden u konstanti (ne vrši se normalizacija mantise);
- CR (count register) sadrži broj cifara iza decimalne tačke;
- SR (sign register) pamti predznak eksponenta (+1 ili -1)
  - Na primer, za konstantu 12.3E4 registri će imati sledeće vrednosti:

NR	ER	CR	SR
123	4	1	+1

1. korak – Konstruisanje prepoznavača

- Ulazna azbuka {DIGIT, •, E, SIGN}
- prelazi razmatranjem karakterističnih primera ulaza

3 8 . 7 E - 3
0 1 1 2 3 4 5 6

. 9 E 2 1
0 7 3 4 6 6

		DIGIT	E	•	SIGN	
inicijalno stanje	0	1		7		0
cifra pre opcione tačke	1	1	4	2		1
opciona decimalna tačka	2	3	4			1
cifra nakon decimalne tačke	3	3	4			1
slovo E	4	6			5	0
predznak eksponenta	5	6				0
cifra eksponenta	6	6				1
tačka iza koje mora ići cifra	7	3				0

2. korak - dodavanje akcija

	DIGIT	E	•	SIGN	⊣
0	1a		7		
1	1b	4a	23c		YES1
23	23a	4b			YES2
4	6a			5	
5	6b				
6	6c				YES3
7	23b				

Akcije imaju sledeći izgled

- Operator val(cifra) daje binarnu vrednost cifre

```
1a:     NR := val(DIGIT)
1b:     NR := NR*10+val(DIGIT)
23a:    CR := CR + 1
        NR := NR*10+val(DIGIT)
23b:    CR := 1
        NR := val(DIGIT)
23c:    CR := 0
7:      ———————
YES1:   ER := 0
        CR := 0
4a:     CR := 0
4b:     ———————
5:      SR := val(SIGN)
6a:     SR := +1
        ER := val(DIGIT)
6b:     ER := val(DIGIT)
6c:     ER := ER * 10 + val(DIGIT)
YES2:   ER := 0
YES3:   ———————
```

- Prepoznavači i procesori
- Implementacija konačnih automata

Varijante implementacije KA

- Pod implicitnom predstavom funkcije prelaza podrazumeva se da je stanje određeno mestom u programu koji se izvršava.
- U eksplicitnoj predstavi, postoji matrica koja pamti funkciju prelaza pri radu automata.
- Objektno-orijentisana implementacija – projektni uzorak **State**.

Primer

- Želi se realizovati na C-u konačni automat sa slike koristeći implicitno i eksplicitno predstavljanje funkcije prelaza.

	a	b	
→A	B		1
B	A	B	0

- Rešenje:

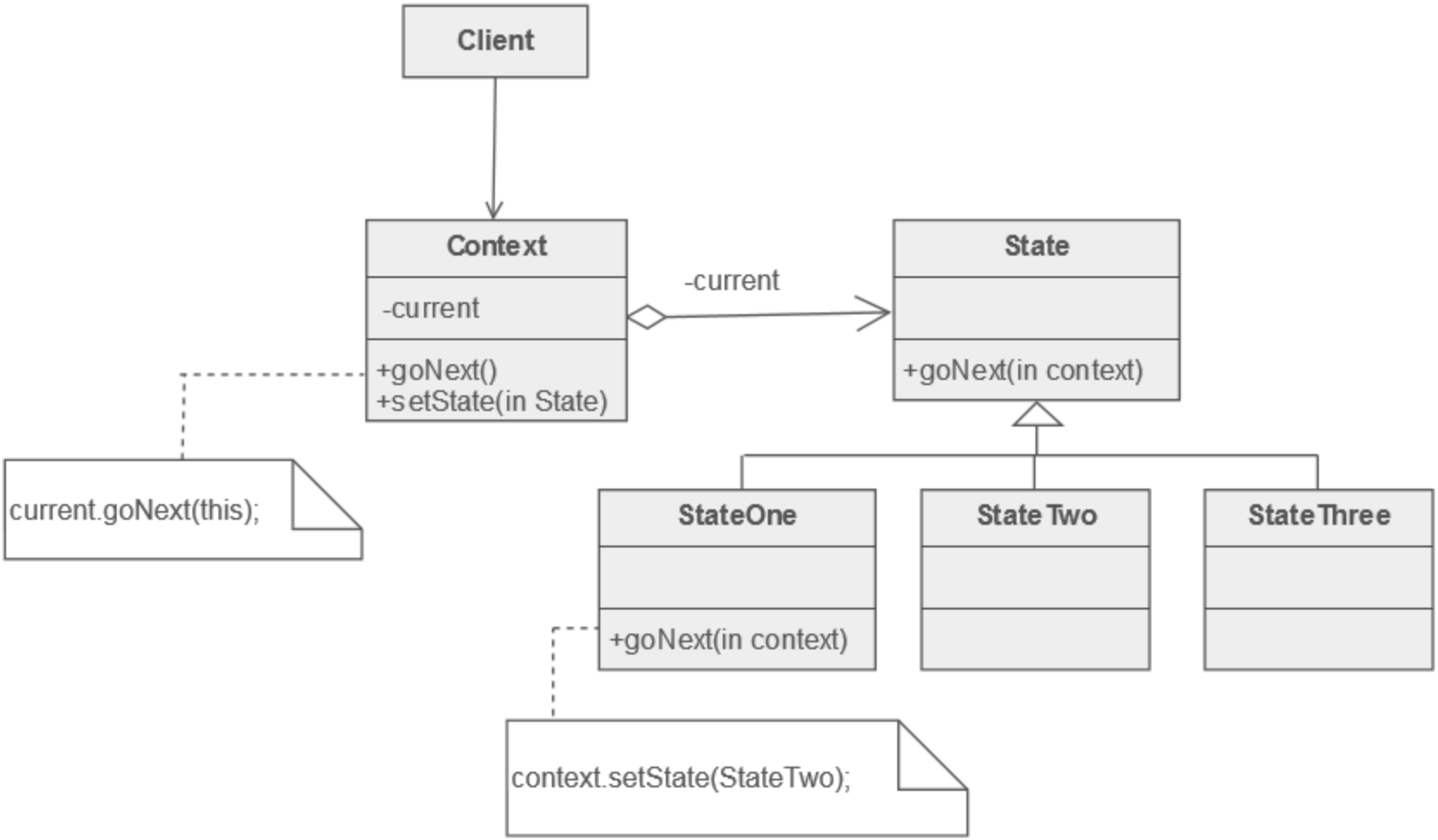
  implicitno FSM.java

  eksplicitno FSM_TT.java

Objektno-orijentisana implementacija KA

- Želimo implementirati zadati automat koristeći projektni uzorak State iz knjige Gama,..."Design Patterns/Elements of Reusable Object-Oriented Programming Style".

Projektni uzorak Stanje



Rešenje: FSM_OO.java